cc-69-7

RADAR (Revised)

by
James W. Meeker

April, 1969

OSU
COMPUTER CENTER
Oregon State University
Corvallis, Oregon 97331

R A D A R

(Revised)

cc-69-7

by

James W. Meeker

April, 1969

Computer Center
Oregon State University
Corvallis, Oregon 97331

RADAR

RADAR* is an advanced, on-line, symbolic debugging system written for the OS-3 Time Sharing system at Oregon State University. RADAR contains an opcode table of the CDC 3300 instructions and a symbol table of programmer-defined symbols which enable it to allow the user to examine and change his program in an assembly language instead of the more machine oriented absolute form. In addition, RADAR has versatile facilities for controlling the execution of a program by both instruction stepping and nine software "breakpoints," with which the user can trace the actual execution of his program.

SYMBOLS:

<SYMBOL>::=Letter|<SYMBOL><letter>|<SYMBOL><digit>|<symbol><period>

Any characters occuring after the eighth one are ignored. Thus, the symbol ABCDEFGH is treated as being the same as the symbol ABCDEFGH765XYZ23PDQ9. Each symbol has a 24-bit binary quantity associated with it called its value. Each symbol must have a value, or it is considered to be undefined. However, two symbols may have values which are equal.

Symbols may be defined in any of three distinct ways. First, symbols may be defined from symbol table output from either COMPASS or FORTRAN by using the SYMLOAD command. Second, if the user types a line of the form

SYMBOL : EXPRESSION

the symbol is put into the table with its value equal to the value of the expression. Third, if the user is typing an expression into an opened word, he may precede the expression with SYMBOL: and the symbol will be defined as having a value equal to the address of the opened word.

*Real-time Assembler-Disassemble and Aid to Research

SYMLOAD:

The symbols used in a COMPASS or FORTRAN program may be loaded into the symbol table if the S option was used at assembly or compilation time, respectively. The command given to RADAR is of the form:

```
SYMLOAD,LUN
or
SYMLOAD,LUN,NAME
```

where NAME is the name of the subprogram and LUN is a general expression which specifies a logical unit number. (The radix is set to ten during evaluation of the unit expression.) If the NAME is specified, only the symbols for that subprogram are processed, otherwise, all symbol records will be processed. The unit specified is initially rewound, and then searched for symbol records with the indicated name, and then rewound when a file mark is encountered. Each time a record is found, the symbols are entered into the symbol table and the values relocated according to information in the LST and RFT of the Loader. Symbols with character relocation and undefined external symbols are not processed.

PURGE:

PURGE followed by a carriage return removes all user defined symbols from the symbol table. The special symbols described below are not affected by this command.

KILL:

This command is used for removing specific symbols from the symbol table. It is particularly useful when the user has a symbol defined which is also an opcode mnemonic (if LDA was defined as a symbol, one could not assemble LDA instructions with RADAR). The format for the KILL command is

```
KILL,symbol
or
KILL,symbol,symbol,...symbol
```

SPECIAL SYMBOLS:

If a "." or "*" occurs when an operand is expected, it is treated as a symbol whose value is the same as the address of the last opened word. The symbols P, A, Q, EU, EL, X1, X2, X3, HIGHMEM, LOWMEM, RADIX, NUMLJ, NUMSS, and NUMSZ are special predefined symbols supplied by RADAR. The values of the first eight symbols are used to store the values of the internal registers when a breakpoint is encountered or whenever RADAR has control of the machine. The values of those eight symbols are restored to the processor registers whenever control returns to the user's program. The value of the symbol RADIX is equal to the current radix of all integers. For example, if RADIX is equal to eight, all numbers are expressed in octal. The last three symbols are used to control the format of integers which are typed out. If the two octal numbers 77777777 and 00000000 are used to represent true and false values of the symbols, then when NUMLJ is true, numbers are printed in left-justified format. When NUMSS is true, the signs are suppressed; and when NUMSZ is true, RADAR suppresses leading zeroes (NUMSZ has no effect if the numbers are left-justified). The values of LOWMEM and HIGHMEM are default values on memory searches and are also modified by the OVREAD and MC commands.

INTEGERS:

Integers are represented internally as one's complement binary integers using 24 bits of precision. Integers may be typed with a space or minus sign or nothing preceding the digits. The digits are packed in the radix specified by the value of the symbol RADIX unless a temporary change has been made as described below.

The radix may be temporarily changed by typing the special character "↑" followed by a B, O, D, or an integer (which is assumed to be in base ten for standardization), and then typing the digits. If the integer is to be negative, the minus sign should precede the up-arrow.

Examples:

```
↑D 11     is 13 in octal
↑O 12     is 12 in octal
↑B 101    is  5 in octal
-↑3 12    is -5 in octal
```

Each number must be terminated by some character that is not a digit. No check is made to be certain that the digits are all less than the radix.

For convenience to systems programmers and other users who work with individual characters, an alphabetic mode has been added. Like the temporary changes of radix described above, the user types "↑" followed by an A. The first character after the A is then treated as a positive 8-bit integer and is the appropriate ASCII code with the most significant bit of the character always being a one bit. An example of this is:

```
↑AX       is 330 in octal.
```

EXPRESSIONS:

An expression is a symbolic string which is used to determine a quantity called the value of the expression. All symbols used in expressions must be defined. The string is evaluated from left to right according to standard precedence rules and conventions concerning the operators and parentheses. The following binary operators are allowed: +, -, *, DIV, MOD, AND, OR, and XOR. In addition, two unary operators, - and @ are allowed. (The @ symbol has the effect of replacing its argument by the word that the address of its argument points to. For example, @ 234 would be equal to the contents of word 234.) An expression may contain any combination of operators and operands so long as the following rules are not violated.

1. Two operands must not be adjacent.
2. A binary operator may not be to the immediate right of any operator.
3. Parentheses must be balanced.
4. The combination @ - is not allowed.

Some examples of legal expressions are:

```
ALPHA DIV 2+3*((-BETA XOR 777)+GAMMA)-@ DELTA+@ (-3 XOR ZORCH)
GAMMA+123
@7654+1
2+2 + 1225DIV (3*ZORRO)
```

The precedence rules are as follows:

1. Quantities within parentheses are evaluated and replaced by their values.
2. @ operators are performed from right to left.
3. Unary minus operators are performed from right to left.
4. AND operators are performed from left to right.
5. OR and XOR operators are performed from left to right.
6. DIV, MOD and * operators are performed from left to right.
7. + and - operators are performed from left to right.

The value of an expression can be printed in the current radix by typing the expression and following it with an equal sign. RADAR will then print out the value and a carriage return with a line-feed. This feature is primarily used to print out the values of symbols, but it can also be used to do integer and logical arithmetic calculations.

ASSEMBLY:

If, in the process of evaluating an expression, an undefined symbol is encountered, RADAR examines the opcode table to see if it is an instruction mnemonic. If it is not, an error has occurred. If it is a mnemonic, the expression evaluator goes into an assembly mode. The basic skeleton of the instruction is set up and RADAR begins assembling the word in almost the same manner as COMPASS, but with the following exceptions:

1. The space is used as a separator, but it does not terminate the field in which it occurs.
2. The comma is used to terminate fields if there is another field which follows. Special characters not having a definite function in assembly terminate the instruction.
3. There are no sub-fields as in COMPASS, except for the opcode modifiers like I, S, EQ, LT, GE, and NE. These fields are assumed to be blank by RADAR unless a comma is the first nonspace character to follow the opcode mnemonic.
4. General expressions are allowed in all fields except for the opcode field and the opcode modifier sub-field.
5. Character addresses are not automatically shifted left two binary places. The user must type "4*" to do the shift.

Thus, the following lines of code may be assembled by RADAR:

```
UJP ZAP

LDA,I 3*(ZORCH+123)AND↑8 7777,BETA XOR 3 +1

LACH 4*ZILCH+6

UJP,I ,3
```

When a blank field has occurred, as in the address field of the last example shown, it is assembled as a zero field.

DISASSEMBLY:

Words being disassembled are printed as an opcode (with its modifier if it has one) followed by all of its fields separated by commas. All fields are printed out as left-justified, sign suppressed integers of the current radix except for the address field. The address field will be expressed as the current location plus or minus an integer less than or equal to seven if it is possible. If that is not possible, then it will be expressed as a symbol plus an integer less than or equal to sixty-three, such that the integer is minimized. If neither of those two are possible, then the address is expressed like any other field. Examples:

```
UJP,I ZORCH+12,0

ENA,S 77777

IJD .-5,2
```

EXAMINATION OF MEMORY:

The examination pointer points to a word which has been opened, that is, a word which may have a new quantity typed into it.

The slash will cause RADAR to type out symbolically (i.e., disassemble) the contents of the location last typed by the programmer or RADAR.

Examples of this are:

```
ADR+3/LDA ZAP+5,1                    (1)

ZORCH/UJP,I P+7,0 /UJP ZORRO,0       (2)
```

If the location which precedes the slash is the first item in the line as in (1), the examination pointer will be set equal to that location. If the location which preceded the slash

is not the first item in the line, as in the second occurrence of (2), then the examination pointer is not changed.

" The quote is the same as slash except that the contents of the location are printed out as integers of the current radix in the format specified by the format control symbols.

The number sign causes the two words at the location to be printed out in floating point. If (CR) is then typed, the location is incremented by two.

' The apostrophe is like the slash except that there is no printout. In addition, the address which preceded the apostrophe is opened, and the quantity which follows is placed in that location. As soon as the quantity is typed, the location specified by the examination pointer is reopened.

= The equal sign causes RADAR to re-type the last quantity in the opposite mode in which it was typed by either the user or RADAR.

Examples:

```
XAPER"012345678

12345'0      (Causes 0 to be placed in location 12345)

SWISH"01073752  =  UJP ZORCH+15,0

BETER/LDA,I BUZZ = 20463572
```

\ The backslash acts just like the apostrophe except that the word which was last typed is opened instead of the word whose address was last typed.

(CR) The carriage return causes RADAR to type a line-feed, increment the examination pointer by one, and print out the new examination pointer followed by a "/", a "'", a "#", or a "'", depending upon the examination mode, followed by the contents of the location, if it is to be printed.

(LF) The line-feed causes RADAR to print a carriage return and a rubout, then set the examination pointer to the last address typed, then proceed as the later part of the carriage return sequence.

; The semicolon causes RADAR to type a carriage return and a line-feed, and to await some input by the user.

BREAKPOINTS:

RADAR has nine breakpoints which allow the programmer to interrupt his program at user-defined points, and to examine the status of the machine from his teletype.

When a breakpoint is encountered in the program, the registers are all saved in the locations which correspond to the values of the symbols which were previously mentioned. The instructions which were originally in the breakpoint locations are restored, and the breakpoint number and the address printed out on the user's teletype. Control then passes to RADAR, which enables the programmer to look around. If the break came from within RADAR, the registers are not saved and control passes to RADAR.

To insert a breakpoint the user need only type $n: EXPRESSION followed by a carriage return. The lower sixteen bits of the expression are then put into breakpoint number n. To find out at what value a breakpoint is set, the user types $n/ and RADAR types out six spaces if the breakpoint is not being used, or the value that the breakpoint is set at if it is in use. After that, the user may type a carriage return, which leaves the breakpoint unaffected, or an expression followed by a carriage return, which causes RADAR to set the breakpoint to the lower sixteen bits of the expression. In either case, RADAR then prints a line-feed.

To remove breakpoints the user types $nK and RADAR deletes breakpoint number n. Typing $K kills all of the breakpoints.

To return to his program after a break has occurred the user types GO and RADAR restores all the registers and saves the instructions at the breakpoint locations, and puts a special instruction in their places. RADAR then jumps to the program after executing the instruction at the breakpoint location from which the break occurred.

RESTRICTIONS ON BREAKPOINTS:

Breakpoints should never be placed in words which are modified or used as data. Neither should they be placed at multiword instructions. No responsibility is assumed if any of these conditions are violated.

EXECUTION OF INSTRUCTIONS:

Any expression may be executed by typing XCT, EXPRESSION followed by a carriage return. RADAR then types a line-feed, followed by one or two extra line-feeds if a single or a double skip occurred. This feature may be used to start a program. For example, to start a program at location ZORCH+3, the user could type XCT, JUMP ZORCH+3 followed by a carriage return.

INSTRUCTION STEPPING:

If a rubout is typed at the beginning of a line, then the instruction at the location pointed to by P will be executed and the value of the symbol P will be changed in the same manner as the hardware program counter. The values of the symbols A, Q, X1, X2, X3, EU, and EL are updated in a similar fashion.

After the instruction has been executed, the instruction at location P is disassembled and a new line is started. More rubouts may be typed and this process repeated. At any point after the first rubout, typing rubout will execute the instruction which was displayed by the previous rubout.

Typing GO(CR) at the start of a line will cause execution to begin at the address indicated by P.

After loading a program with the loader the user may type RADAR instead of RUN. This causes the value of the primary transfer symbol to be stored in P and the value of the secondary transfer symbol to be stored in Q. Control is then passed to RADAR main control. The user may now change words, examine memory, set breakpoints, and so forth. Execution of the program will begin if the GO statement is used.

At the beginning of any line the user may find out what the contents of the dynamic registers are by typing STATUS followed by carriage return. If the user is relocated to operand state, then [ROS] is printed; otherwise, this line is absent. The names of the registers followed by the contents expressed in octal are printed on successive lines. The STATUS command may also be used to determine the status of a logical unit. The command is in the form STATUS,LUN followed by a carriage return. LUN is a general expression evaluated in base ten. The command prints out the hardware type followed by the names of the status bits.

BREAK:

The system control statement BREAK causes an interrupt to RADAR's break location. This is useful when a program gets into a loop, or if the user causes an illegal instruction within RADAR itself. The latter case will cause RADAR to enter main control and not disturb the user's registers since the break originated from within RADAR.

RADAR:

The system control statement RADAR causes RADAR to be loaded into the user's virtual memory and control passed to RADAR main control.

LOGICAL UNIT FUNCTIONS:

A function may be applied to a series of logical units with a command of the form:

FUNCTION,LUN,LUN,LUN,...LUN

where LUN is a general expression which is evaluated with radix equal to ten, and FUNCTION is one of the following:

BACKSPACE	backspace each unit.
BKSPACE	same as BACKSPACE.
BKSP	same as BACKSPACE.
CLEAR	clear status on each unit.
FWDSPACE	forward space on each unit.
FWSP	same as FWDSPACE.
RELEASE	release each unit.
REWIND	rewind each unit.
SBPFM	search backward past file mark on each unit.
SEFB	same as SBPFM.
SEFF	same as SFPFM.
SFPFM	search forward past file mark on each unit.
UNEQUIP	unequip each unit.
WEOF	same as WFM.
WFM	write file mark on each unit.

In addition, RADAR processes EQUIP commands in the same format as those processed in system control mode.

SEARCHES:

The user may search memory for any quantity looking at only certain bits which are user defined. The search is similar to that which the MEQ instruction does, but unlike the hardware masked equality search, the user may look for 77777777 octal and not find a lot of zero words, too. To start a search, the user types:

SRCH,EXPRESSION:EXPRESSION:EXPRESSION:EXPRESSION

followed by a slash, an apostrophe, or a quote mark. When

the search finds a word it prints the address followed by the symbolic contents of the word if slash was typed, or the numeric contents of the word if a quote was typed, or nothing if the apostrophe was typed. The first and second expressions typed in setting up the search are the lower and upper limits of the search respectively. The value of LOWMEM is assumed for the first expression if the expression is empty, and the value of HIGHMEM is assumed for the second expression if it is empty. The value of the third expression is the quantity being searched for, and the last expression is the mask. In the search, bits are compared only where there are one bits in the mask. Thus,

SRCH,ZAP:ZORCH+5:ADR:77777/

will have RADAR search memory from ZAP to ZORCH+5 looking for ADR in the lower fifteen bits of each word and ignoring the upper nine bits of each word that it examines. If it finds some words it might type:

65234/UJP ADR,0

76702/UJP,I ADR,1

MC:

The command MC followed by a carriage return causes RADAR to zero all the dynamic registers, clear all the breakpoints, clear the ROS condition, and reinitialize LOWMEM and HIGHMEM to 00000 and 77777_8 respectively. The command MC,I/O followed by a carriage return has the effect of unequipping all logical units and then equipping 60 and 61 to the standard terminal input-output.

OVERLAYS:

Two commands are supplied by RADAR to facilitate making minor corrections to overlays. Both commands rewind the unit

specified before performing the operation and OVWRITE will equip the unit as a file if it is not already defined. OVREAD reads the header record into the lower four words of memory and then uses the information in the header as fifteen bit addresses to read the overlay record into the lower 32K of virtual memory. OVREAD also sets the program counter and Q register appropriately. OVWRITE uses the lower four words of core in an inverse fashion. Format for these commands is as follows:

```
OVREAD,LUN
and
OVWRITE,LUN
```

each is terminated by a carriage return and LUN is a general expression evaluated in base ten arithmetic.

ZEROCORE:

The ZEROCORE command is terminated with a carriage return, and its effect is to set the lower 32K of virtual memory to zeroes.

MISCELLANEOUS:

To change the format or radix of numeric printout, the user defines the special symbols previously shown. The RADAR system is initialized as being octal, sign-suppressed, right-justified, and printing leading zeroes. To change to decimal sign printing, the user would type:

```
RADIX ← ↑D 10
NUMSS ← 0
```

Since the two most frequent modes of printout require three symbol definitions to switch back and forth, two commands, DECIMAL and OCTAL have been implemented which change the mode to sign-printing, left-justified decimal or sign-suppressing, right-justified octal respectively.